A Cascade Model for Proposition Extraction in Argumentation

Jo, Yohan; Visser, Jacky; Reed, Chris; Hovy, Eduard

Published in:
Proceedings of the 6th Workshop on Argument Mining

DOI:
10.18653/v1/W19-4502

Publication date:
2019

A Cascade Model for Proposition Extraction in Argumentation

Yohan Jo[1], **Jacky Visser**[2], **Chris Reed**[2], **Eduard Hovy**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]Centre for Argument Technology, University of Dundee
yohanj@cs.cmu.edu, j.visser@dundee.ac.uk,
c.a.reed@dundee.ac.uk, hovy@cmu.edu

Abstract

We present a model to tackle a fundamental but understudied problem in computational argumentation: proposition extraction. Propositions are the basic units of an argument and the primary building blocks of most argument mining systems. However, they are usually substituted by argumentative discourse units obtained via surface-level text segmentation, which may yield text segments that lack semantic information necessary for subsequent argument mining processes. In contrast, our cascade model aims to extract complete propositions by handling anaphora resolution, text segmentation, reported speech, questions, imperatives, missing subjects, and revision. We formulate each task as a computational problem and test various models using a corpus of the 2016 U.S. presidential debates. We show promising performance for some tasks and discuss main challenges in proposition extraction.

1 Introduction

Most argument mining models for identifying the argumentative structure of a text build upon elementary text spans that serve argumentative functions, such as premise and conclusion. In argumentation theory, it is commonly accepted that these building blocks are propositions (Blackburn, 2016), i.e., statements that are either true or false. Despite the foundational role of propositions, however, proposition extraction from text has been little studied in computational argumentation. Instead, most models rely on argumentative discourse units (ADUs) obtained by surface-level text segmentation (Stede et al., 2016; Al Khatib et al., 2016). In what follows, we discuss limitations of ADUs that potentially impinge upon subsequent argument mining processes, and then describe our approach.

One limitation of ADUs is that they may lack important semantic information, such as the **referents of anaphors** and the **subject of an incomplete sentence**, necessary for subsequent argument mining steps. For example, for two consecutive text segments *Alice complained to Bob* and *He is upset*, if we do not know *he* refers to Bob, it would be confusing whether the first segment supports the second or vice versa. In another example, suppose *Alice was faithful to Bob, keeping the secret* is split into two propositions, each associated with the main clause and the adverbial participle, respectively. While mere text segmentation leaves the subject of the participle (Alice) missing, tracing and reconstructing the subject makes it clear that the participle supports the main clause. As illustrated in these examples, anaphora resolution and subject reconstruction recover semantic information that has potential benefits for argument mining systems.

Moreover, ADUs may completely miss implicit propositions. For instance, **questions** and **imperatives** do not convey explicit propositions, but they are important argumentative components that often imply propositional content in dialogical argumentation. Suppose an arguer asks, *why would you waste your money on tax?*, and someone responds, *tax is a waste of money*. It is not straightforward for an argument mining system to tell whether the response agrees or disagrees with the arguer, without knowing what is implied by the question. Implicit propositions occur in **reported speech** as well. Suppose an arguer says, *the doctor said we need more magnesium*. The arguer is not only claiming the report event having happened, but also bringing the content of the doctor's speech as a proposition into the argumentation structure or even may be asserting it using authority. These examples show the significance of recovering implicit propositions for argument mining systems.

To overcome these limitations, we present a cascade model that aims to extract propositions

Proceedings of the 6th Workshop on Argument Mining, pages 11–24
Florence, Italy, August 1, 2019.

from argumentative dialogues, with important semantic information and implicit propositional content recovered. Our model consists of 7 modules, namely, anaphora resolution, locution extraction, reported speech, question, imperative, subject reconstruction, and revision (Figure 2). For each module, we formulate the task as a computational problem and test various models to solve it, except for the question and imperative modules, for which we present experimental sketches. Our analyses and evaluation are based on the transcripts of the 2016 U.S. presidential debates and reaction on social media that are manually annotated with propositions (Visser et al., 2019). Our contributions are three-fold.

1. We introduce the problem of proposition extraction as seven tasks.
2. We present various models to tackle each task and evaluate performance.
3. We analyze challenges facing our computational methods and suggest future directions.

For the remainder of the paper, we first review prior work on ADU segmentation and a theoretical framework for obtaining propositions from ADUs (§2). We then explain the annotated data of propositions (§3). Next, we describe our cascade model (§4) and formulation of each task, along with experiments (§5). We conclude the paper by discussing the challenges and future directions (§6).

2 Background

In computational argumentation, the basic unit of an argument is often called an argumentative discourse unit (ADU). In this section, we first review how existing studies define and obtain ADUs from text, and then some theoretical framework to obtain propositions from ADUs.

2.1 From Text to ADUs

In most studies, ADUs are obtained via text segmentation. While some studies leave the choice of the boundary of an ADU to the annotator's judgment (Stab and Gurevych, 2014), many studies employ a set of syntactic rules as a basis. For instance, an ADU can be as fine-grained as a phrase that plays a discrete argumentative function (Stede et al., 2016). In other cases, an ADU may be a clause (Peldszus and Stede, 2015) or a series of clauses that must include a subject, a verb, and an object if necessary (Al Khatib et al., 2016).

Based on annotated ADUs, some studies have proposed methods for automatically segmenting ADUs using machine learning. This task is commonly formulated as tagging each word in the text as either the beginning, inside, or outside of an ADU (BIO tagging). The tagging has been incorporated into an end-to-end argument mining (Eger et al., 2017) or conducted separately on various domains (Ajjour et al., 2017). Instead of tagging, a retrieval approach has also been used, where candidate ADUs are generated and the best is retrieved (Persing and Ng, 2016).

All these approaches to ADU segmentation share most of the concerns mentioned in Section 1. For better-informed argument mining, we need to go further to obtain propositions from ADUs, and thus a relevant framework will be discussed in the following section.

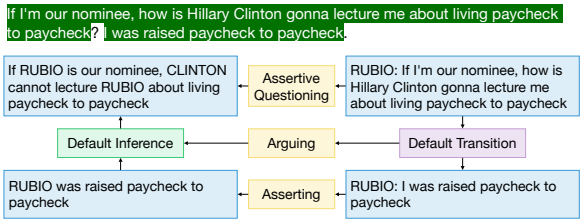


Figure 1: A snippet of the US2016 corpus. The top text is the original utterance. The blue boxes on the right are locutions, which are also highlighted with green on the utterance. The blue boxes on the left are propositions anchored in the locutions, via illocutionary acts (yellow boxes).

2.2 From ADUs to Propositions

Following Speech Act Theory (Austin, 1962; Searle, 1969), the connection between text segments and propositions can be modeled as illocutionary acts: the application of particular communicative intentions to propositional contents – e.g., *asserting* that a proposition is true, or *questioning* whether it is true. Focusing on argumentatively relevant speech acts (van Eemeren and Grootendorst, 1984), Inference Anchoring Theory (IAT) (Reed and Budzynska, 2011) explains how propositional contents and the argumentative relations between them are anchored in the expressed locutions by means of illocutionary connections.

IAT has been applied to annotate argumentative dialogues of various kinds, including the corpus used in this paper (Section 3). IAT annotation comprises, amongst other things, segmenting the original text into locutions[1], identifying the illo-

[1] Analogous to ADUs. We use the terms interchangeably.

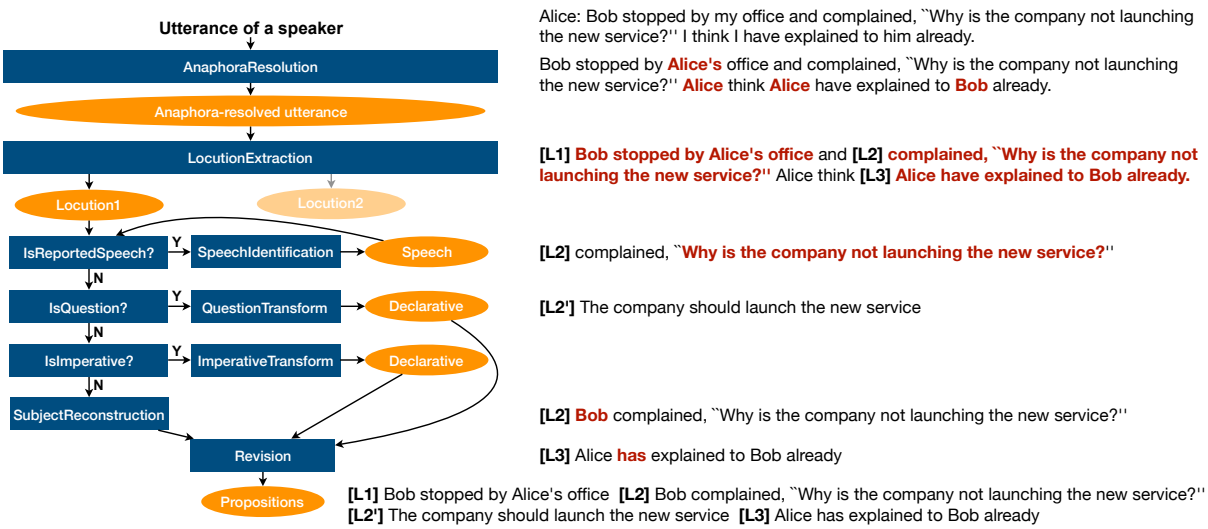


Figure 2: Cascade model of proposition extraction. The input is each utterance, blue boxes are individual (sub)modules and orange circles are the outputs of the modules. We made up the utterance used in the figure in order to cover the functions of most modules.

cutionary force instantiated by the locution, and reconstructing its propositional content (an example snippet shown in Figure 1). Each locution generally conveys a propositional content. Conjuncts conjoined by a conjunction and conditional clauses may be separated if they each fulfill a discrete argumentative function. In addition, punctuation, discourse indicators, and epistemic modalities (e.g., *I think*) should be excluded. For propositions, anaphoric references are typically reconstructed, resulting in full grammatical sentences understandable without context.

3 Data

We use the US2016 corpus (Visser et al., 2019), which contains transcripts of televized debates for the 2016 U.S. presidential election and reaction to the debates on Reddit. All dialogues have been manually segmented and annotated with locutions, illocutionary connections, and propositions based on IAT (Reed et al., 2016) (Figure 1). The corpus was annotated by 4 annotators, yielding an overall Cohen's κ of 0.610 (considered substantial agreement). We downloaded the annotations from the corpus webpage and separately scraped the original dialogues.

For data preparation, we aligned each locution with the original dialogue; e.g., in Figure 1, the locutions (in the right blue boxes) are aligned with the original utterance (at the top) using text matching. This allows us to build a model to extract locutions from utterances, and propositions from locutions. As our model handles reported speech and questions, we need additional processing to identify those locutions. In the corpus, a locution of reported speech (e.g., *S said P*) is annotated with an intermediate locution, along with the speaker (*S*) and the content of speech (*P*). The content of speech, in turn, becomes the proposition of this locution. Locutions of questions are connected with their propositions via four illocutionary acts: pure/assertive/challenge/directive questioning. The processed data includes 2,672 utterances and 8,008 locutions (278 reported speech and 565 questions).

4 Model

Our cascade model takes a speaker's utterance as input, runs seven modules, and outputs a set of propositions extracted from the utterance. Figure 2 shows the model structure and an example utterance processed throughout. The functions of individual modules are as follows:

1. **Anaphora resolution:** Replace pronoun anaphors with their referents.
2. **Locution extraction:** Extract locutions (ADUs) from the utterance.
3. **Reported speech:** Determine if the locution is reported speech; if so, identify the text segment representing the content of speech.
4. **Question:** Determine if the locution or speech content is a question; if so, extract its propositional content.
5. **Imperative:** Determine if the locution or

speech content is an imperative; if so, extract its propositional content.

6. **Subject reconstruction:** Reconstruct the missing subject, if any, of the locution or speech content.
7. **Revision:** Make additional adjustments necessary for final propositions.

5 Method

In this section, we describe how to formulate the task of each module as a computational problem, and present various approaches with their performance. Each module is evaluated separately on the ground truth data, instead of using the result of the previous module. This setting prevents error propagation and helps to evaluate the performance of each module more accurately. Some methods we use are based on machine learning and thus requires a split of training and test sets. Hence, we randomly split the entire corpus into five folds and conduct cross validation with the same folds throughout the paper.

Extensive experiments are focused on anaphora resolution, locution extraction, subject reconstruction, and revision. For the other modules, we present baseline models or experimental sketches, leaving room for improvement for future work.

5.1 Module: AnaphoraResolution

Anaphora resolution is based on Stanford CoreNLP 3.8.0. Yet, blindly applying it induces several challenges, such as incorrect resolution of speakers/hearers (this information may be often missing in the text), resolution of non-pronouns, and errors inherent in the tool. To rectify these challenges, we decompose the task into the following subtasks.

- **1st-person singular:** Replace *I*, *my*, *me*, *mine* with the speaker's name.
- **2nd-person singular:** Replace *you*, *your*, *yours* with the previous turn's speaker name.
- **3rd-person singular gender:** Resolve *he, his, him, she, her, hers* using CoreNLP.
- **3rd-person singular gender-neutral:** Resolve *it, that* using CoreNLP.
- **3rd-person plural:** Resolve *they, their, them, theirs* using CoreNLP.

Inaccurate anaphora resolution can rather distort the original meaning of text. Hence, the goal here is to find the best combination of the subtasks. The first two subtasks are applied only to TV debates, as Reddit user names have not been resolved in the corpus. All possessive pronouns are replaced with references suffixed with *'s* (e.g., *his* $\rightarrow$ *Trump's*).

	BLEU	Dep	Dep-SO	Noun
Locution (no resol)	69.3	.651	.558	.714
CoreNLP	62.8	.617	.538	.704
1S	**70.1**	**.657**	.589	.748
1S+2S	69.7	.655	.583	.746
1S+3SG	69.3	.654	**.601**	**.757**
1S+3SG+3SN	68.5	.649	.592	.756

Table 1: Performance of anaphora resolution. (**1S:** 1st-person singular, **2S:** 2nd-person singular, **3SG:** 3rd-person singular gender, **3SN:** 3rd-person singular gender-neutral, **Dep:** Dependency, **Dep-SO:** Dependency for subjects and objects.)

For evaluation, we assume that effective anaphora resolution would make a locution more "similar" to the annotated proposition. Hence, we compare the similarities between a locution and the annotated proposition before and after anaphora resolution, using the following metrics:

- **BLEU:** Generic string similarity based on n-grams ($n = 1, 2, 3, 4$).
- **F1-score of dependency tuples:** String similarity based on dependencies. Less sensitive than BLEU to the exact locations of words.
- **F1-score of nsubj/dobj dependency tuples:** Rough semantic information pieces representing who did what to whom/what.
- **F1-score of nouns:** How accurately anaphora resolution retrieves nouns (as our anaphora resolution replaces only nouns).

Result

As shown in Table 1, blindly applying CoreNLP (row 2) significantly hurts all similarity measures (compared to row 1). In contrast, speaker resolution (row 3) plays a key role in improving all measures over original locutions, especially semantic information (subject/object) and nouns. Additional resolution of hearers (row 4) does not help, as *you* is used in a more general way than referring specifically to the hearer.

Resolving 3rd-person gender pronouns (row 5) further improves performance for semantic information and noun retrieval over speaker resolution, at the expense of slightly lower BLEU and dependency similarites. Additional resolution of *it*, *its*, and *that* turns out to rather hurt performance.

For argument mining, it may be desired to resolve as many anaphors as possible unless the

original meaning is significantly hurt, because pronouns provide little information for identifying propositional relations. Hence, we conclude that resolution of speakers and 3rd-person gender pronouns is ideal for this module, and the subsequent modules use the result of this configuration. However, we find that resolution of 3rd-person gender-neutral pronouns is critical, as will be discussed in Section 5.8, and eventually they should be resolved depending on the availability of proper anaphora resolution tools.

5.2 Module: LocutionExtraction

For each utterance with anaphors resolved, the LocutionExtraction module identifies locutions, from which proposition(s) will be extracted. This task is almost identical to ADU segmentation, and several methods have already been proposed (Section 2.1). Beating prior models for this task is beyond the scope of this paper; rather, we focus on understanding what causes confusion for locution boundaries. Following the convention for this task (Eger et al., 2017; Ajjour et al., 2017), the task is formulated as tagging each word with B/I/O (beginning/inside/outside of a locution).

We explore the state-of-the-art BiLSTM model (Ajjour) (Ajjour et al., 2017), as well as a regular CRF (R-CRF) and BiLSTM-CRF (Huang et al., 2015). A CRF showed strong performance for cross-domain segmentation, and BiLSTM-CRF is an extension of CRFs, where emission scores are calculated through BiLSTM. For all models, we use the following features, adopted from or informed by the prior work (Ajjour et al., 2017):

- **word:** Current word (i.e., word index for R-CRF and pre-trained GloVe.840B.300d word embeddings for BiLSTM-CRF and Ajjour).
- **pos:** Part-of-speech tag of the current word.
- **ne:** Named entity type of the current word.
- **prev_1gram:** Previous word of the current word, as conjunctions and discourse markers are good indicators of locution boundaries. (R-CRF only, as BiLSTM considers context.)
- **bos/eos:** Indicator of whether the current word marks the beginning/end of a sentence, as locution boundaries are often restricted by sentence boundaries.
- **boc/eoc:** Indicator of whether the current word marks the beginning/end of a clause, as locution boundaries are closely related to clause boundaries. We obtain clauses from the constituency parse of the sentence, taking phrases tagged with S. For nested clauses, we take the deepest clauses to avoid overlap.

The model settings are explained in Appendix A.

We evaluate the models using the macro F1-score across the BIO tags with 5-fold CV.

R-CRF	BiLSTM-CRF	Ajjour
.788	.789	**.794**

Table 2: F1-score of locution extraction.

	1st locution1	2nd locution
Subordinate clauses	7%	6%
Adverb phrases	4%	8%
Particle phrases	1%	4%
Yes/no	2%	-
Relative clauses	-	5%

Table 3: Breakdown of locution types that are separated by a comma or that are back-to-back (total 293 pairs).

Result

Ajjour et al. (2017)'s model outperforms the CRF-based models (Table 2). The model tends to underproduce locutions (7,767 compared to 8,008 annotated), i.e., produce coarse locutions, missing signals for splitting them further into smaller locutions. To examine those signals, we gathered extracted locutions that overlap with two consecutive annotated locutions, and counted the words between the two locutions (Table 9 in Appendix).

Frequently, the model failed to make a split at a comma (31%) or between locutions that are back-to-back without any separator in between (10%). In the majority of these cases, the locutions are two independent clauses, indicating that the model needs a more robust mechanism to make use of clause boundaries. Although not very common, a locution also serves as a subordinate clause, adverb phrase, particle phrase, yes/no answer, or relative clause (Table 3). Deciding whether to separate a subordinate clause from the main clause is not trivial. For instance, if- and when-clauses, the most common subordinate clauses in the analysis, are separated off or attached to the main clause depending on the strength of their dependency, which is often vague. If we are to build a system to make this decision automatically, we may consider the truth value of the subordinate clause and whether it is idiomatic.

Other frequent separators include conjunctions *and* (21%) and *but* (6%). As in the case above, the

Regex	Prec	Recl	F1
say + said	.404	.363	.383
Reporting marks	.576	.259	.357
Other reporting verbs	.579	.040	.074
All above	.442	.590	.505

Table 4: Accuracy of reported speech detection.

model sometimes has difficulty deciding whether to split conjoined phrases and clauses.

5.3 Module: ReportedSpeech

A locution extracted above is examined by the IsReportedSpeech submodule to decide if it is reported speech. If so, the content of speech is identified by the SpeechIdentification submodule.

5.3.1 Submodule: IsReportedSpeech

To detect if a locution is reported speech, we use 11 regular expressions that capture the existence of reporting verbs (*said*, *say*, *called*, *blamed*, *argued*, *insisted*) and reporting marks (*“*, *:*). A matched locution is classified as reported speech.

Result

As shown in Table 4, the method achieves an F1-score of 0.505, which reveals the difficulty of detecting reported speech (the full list of patterns and their accuracy are in Table 10 in Appendix). High-performing patterns capture *say*/*said* and reporting marks; other reporting verbs have too low recall. Interestingly, regular expressions achieve not only low recall but also low precision. To see why, we examined false-positives made by *said* and opening quotation marks, and found this task quite challenging indeed. Two big challenges are detecting whether the report actually happened and if the content of speech is mentioned, as in the following examples (underlined text increases complexity):

1. **Event factuality:** *I thought reddit said that Paul was supposed to be the rational one here*; *He never even said that he didn't do it*
2. **Mention of speech content:** *He said that the second time anyway*; *I mean, “track the terrorists and not the citizens” is full of so many holes*

These challenges suggest that we need more sophisticated features to identify event factuality and the mention of speech content.

Tregex	F1	Coverage
Reporting verbs	.234	5%
Reporting marks	.371	20%
All above	.395	23%

Table 5: Accuracy of speech identification.

5.3.2 Submodule: SpeechIdentification

Speech content is important to identify, as it often contributes to the argumentation structure (e.g., as part of an authority claim). We formulate this task as BIO tagging, as in locution extraction. Individual words in each locution are tagged with B/I/O based on the best alignment between the locution and its content proposition (Section 3).

To identify speech content, we use regular expressions matched to constituency parse trees. A speech is assumed to be a clause, preceded by a reporting verb (*said*, *say*, *says*, *claim.**, *argue.**, *insist.**) or reporting mark (*“*, *:*). Matching is conducted using Tregex in Stanford CoreNLP.

For evaluation, a matched clause is tagged with B and I, and the other words with O. We compute the macro F1-score of BIO tags and the percentage of locutions matched by patterns (coverage).

Result

As shown in Table 5, the Tregex method has a low F1-score, mostly due to a poor coverage (the full list of patterns and their accuracy are in Table 11 in Appendix). The low coverage stems from several causes, including:

- Speech content may not be a complete clause (e.g., *you say charge the banks more*), or the parser fails to recognize it as a clause.
- Speech content is signaled by various verbs (e.g., *talking about*, *I'm hearing*).
- A reporting verb may be missing. This usually happens when speech content and main clause are segmented into separate locutions.

We believe that various signals of speech content may be captured better by machine learning models, increasing overall performance.

5.4 Module: Question

A locution or speech content is examined by the IsQuestion submodule to decide if it is a question. If so, it is transformed to its propositional content by the QuestionTransformation submodule.

Regex	Prec	Recl	F1
Question mark	.751	.938	.834
Initiating words	.514	.499	.506
All above	.588	.972	.733

Table 6: Accuracy of question detection.

5.4.1 Submodule: IsQuestion

To detect if an input text is a question, we use regular expressions that capture if the enclosing sentence has a question mark or begins with words that often initiate a question (e.g., *how*, *do*) (the full list of patterns is in Table 12 in Appendix). The reason for matching the patterns to the entire sentence is that a question mark is often excluded from a locution.

Result

As shown in Table 6, a question mark by itself is strongly indicative of a question and has high recall. While it has fair precision, there exist some confusing false-positives, including:

1. A question merely for emphasis. (e.g., *It also could be somebody sitting on their bed that weighs 400 pounds, ok?*)
2. Reported question. (e.g., *You say to yourself, why didn't they make the right deal?*)
3. A question for expressing confusion. (e.g., *Bernie?... Come again?*)

For questions without a question mark, the regular expressions for question-initiating words increase recall but significantly hurts precision. Some of these words are used as a subordinate conjunction (*when*) or as a relative pronoun (*which*). The low precision of some words is due to incomplete sentences with missing subject *I* (e.g., *Could barely understand*). The error cases show that highly accurate detection of a question requires a combination of several factors.

5.5 Submodule: QuestionTransformation

We found no prior work that addresses transforming questions into propositions, although some work identifies different types of questions (Zhang et al., 2017). For this submodule, we only describe the task with examples in the corpus, without models. In the corpus, questions are associated with four illocutionary acts: pure, assertive, challenge, and directive. Pure questions assume no assertion and thus may be transformed to a statement underspecified in the semantic dimension questioned, optionally containing a placeholder *xxx*:

- *Who is Chafee?* → *Chafee is xxx*
- *Do all lives matter?* → *All lives do / do not matter*

In contrast, assertive and challenge questions have an assertive force. The difference between them is whether or not a question is to challenge another argument.

- *What does that say about your ability to handle challenging crises as president?* → *Clinton does not have the ability to handle challenging crises as president.* (assertive)
- *What has he not answered?* → *He has answered questions* (challenge)

Lastly, directive questions have imperative mood:

- *Any specific examples?* → *Provide any specific examples*

We may explore various approaches, such as handcrafted rules and seq2seq models, in future work.

5.6 Module: Imperative

There is neither consensus nor common practice on how to extract propositional content from imperatives. Accordingly, the corpus had no guidelines for imperatives, and most imperatives have not been modified. Yet, some imperatives have been modified according to the annotators' own judgment, with examples including:

- *Raise the minimum wage* → *The minimum wage should be raised*
- *Let me address college affordability* → *Clinton would like to address college affordability*
- *Look at the mess we're in* → *We're in a mess*

We argue that more analysis would be useful to understand when and how an imperative can be systematically transformed to a proposition.

5.7 Module: SubjectReconstruction

A locution or speech content may miss its subject due to segmentation. Hence, the SubjectReconstruction module aims to reconstruct the subject if it exists within the same sentence. We first trace the subject of each verb in every sentence, and then reconstruct the subject (along with auxiliary verbs) of a segmented text that begins with a verb whose subject is outside the text.

We trace the subject of a verb using the basic dependency relations (from CoreNLP) as follows. When a verb has no subject relation with any words, we move to the word that is connected with the current verb through a dependency relation of the types: conjunct (conj), auxil-

Prec	BLEU-Reconst	BLEU-Locution
.714	62.6	59.1

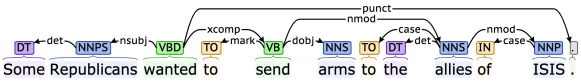
(a) Performance of subject reconstruction.

Reason	%
Ill-formed sentence	25%
No subject in the sentence	25%
Trace mistake	20%
Complex sentence	10%
Phrasal/clausal subject	10%
Wrong antecedents of relative pronouns	10%

(b) Reasons for subject identification errors.

Table 7: Results of subject identification.

iary (aux/auxpass), copula (cop), and open clausal complement (xcomp). The intuition is that this new word and the current word are likely to have the same subject. We repeat this process until we find a subject or no more move is available. The following dependency parse illustrates the intuition, i.e., why *wanted* and *send* connected with xcomp have the same subject. Examples of the other relations are in Appendix B.



Sometimes a verb's direct subject is a relative pronoun, in which case we move to the word modified by the verb via the acl:relcl relation. However, *which* may often refer to a phrase or a clause, and this method may not be able to capture that.

Result

We identified 96 locutions (1.2% of locutions) beginning with a verb whose subject is identified to be in the sentence yet outside the locution. We focus on 73% of them whose subjects are recovered in annotated propositions. Note that annotated subjects can be lexically different from the ones that are correctly identified by our method, due to imperfect anaphora resolution. Hence, our evaluation is based on manual comparison, checking if identified subjects and annotated subjects refer to the same thing/person.

As shown in Table 7a, the method identified subjects correctly for 71% of the locutions. Accordingly, the BLEU score improved by 3.5, compared to mere locutions. Table 7b breaks down the reasons for errors. Sometimes the tracing method made a mistake (20%) or failed to capture a phrasal/clausal subject (10%). However, more commonly, CoreNLP could not properly handle sentences that are ill-formed (25%), missing a subject (25%), or too long/complex (10%). In some cases, it incorrectly identified the antecedents of relative pronouns (10%).

There exists other work that addresses recovering elided materials in sentences using dependencies (Schuster et al., 2018). Following some of the work, it would be an interesting direction to explore a richer set of dependency relations, such as the enhanced dependencies (Schuster and Manning, 2016).

5.8 Module: Revision

While the previous modules handle major tasks, a processed locution may still need additional adjustments, including grammar correction. Hence, the Revision module makes adjustments to a processed locution and outputs proposition(s). This task is formulated as a seq2seq problem, i.e., a model automatically learns and decides how to change the input, based on the data.

We explore two models: standard attention (Luong et al., 2015) and copy mechanism. Both encode an input text using BiLSTM and decode proposition(s) using LSTM. The attention model computes the probability of a word being generated, using attention over the encoder's hidden states. It requires a lot of training data, whereas we already know that most input words remain unchanged. The copy model, on the other hand, decides internally whether to copy an input word or generate a new word. Informed by existing copy mechanisms (Gu et al., 2016; Allamanis et al., 2016), we developed a slight variant that worked better on this task. The model and parameters are explained in detail in Appendix C.

We use two evaluation metrics: BLEU and exact match (percentage of outputs identical to the annotated propositions). We exclude locutions of reported speech and questions, to better focus on this module's performance. The baseline is to treat each locution as a proposition without modification. Accuracy is based on 5-fold CV.

Result

As shown in Table 8, the baseline (row 1) already achieves high performance, because locutions are often very similar to the propositions extracted from them unless they are reported speech or questions. For this reason, the attention model (row 2) performs poorly, as it tends to make many

	BLEU	Exact
Locution	75.5	.473
Attention	47.2	.124
Copy	76.2	.493
Copy (short)	**76.6**	**.501**

Table 8: Performance of revision.

unnecessary adjustments to input locutions. The copy model (row 3) performs significantly better than the attention model, but sometimes it could not handle long input texts and generated irrelevant content toward the end of an output. Leaving long input texts (25+ words) unmodified (row 4) slightly improved performance. Overall, the improvement over the baseline is rather modest.

The most notable and useful role of the copy model is correcting a verb case that was left incorrect due to anaphora resolution (e.g., *cooper want to* → *cooper wants to*, *webb have had* → *webb has had*). This behavior is quite desirable. The model also sometimes removed non-propositional content and changed a person's first name to the full name as reflected in annotations. In general, the roles of the model remain lexical conversion rather than semantic conversion.

We found that the differences between generated and annotated propositions are derived mainly from unresolved non-personal anaphors (e.g., *it*, *this*, *that*). Furthermore, annotators sometimes insert omitted verb phrases (e.g., *You should.* → *You should clinge to capitalism.*; *not hard to do* → *not hard to dominate*). Such semantic information is not recovered by the current copy model.

6 Conclusion

Our decomposition of the proposition extraction task has yielded that: (i) anaphora resolution is crucial for recovering the semantic information of propositions, and the main bottleneck is to resolve 2nd-person singular and 3rd-person gender-neutral pronouns; (ii) locution boundaries are often confused around clause boundaries; (iii) detecting reported speech and speech content suffers poor accuracy with pattern matching. These tasks, along with question detection, reveal the need for sophisticated feature combinations for satisfactory results, and we may need additional training data; (iv) for subject reconstruction, the tracing method is fairly effective, and the accuracy is bounded mainly by the robustness of dependency parsing to ill-formed and complex sentences; (v) the final revision with a seq2seq model remains mostly grammar error correction, and substantial semantic revision may require significantly different models.

Though we are starting to explore the challenges facing complete reconstruction of propositions from natural argumentative discourse, our cascade model already demonstrates improvement over locutions (ADUs) in several modules for this understudied yet crucial task in argument mining.

We are currently working on systematic extraction of propositional content from questions and imperatives, and evaluation of the entire cascade model as a whole. Our future direction is to use extracted propositions to develop argument mining models that identify nuanced types of propositional relations informed by argumentation theory.

Acknowledgements

This research was funded by the Kwanjeong Educational Foundation and grant EP/N014871/1 by the Engineering and Physical Sciences Research Council (EPSRC).

References

Yamen Ajjour, Wei-Fan Chen, Johannes Kiesel, Henning Wachsmuth, and Benno Stein. 2017. Unit Segmentation of Argumentative Texts. In *Proceedings of the 4th Workshop on Argument Mining*, pages 118–128, Copenhagen, Denmark. Association for Computational Linguistics.

Khalid Al Khatib, Henning Wachsmuth, Johannes Kiesel, Matthias Hagen, and Benno Stein. 2016. A News Editorial Corpus for Mining Argumentation Strategies. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3433–3443. The COLING 2016 Organizing Committee.

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A Convolutional Attention Network for Extreme Summarization of Source Code. In *International Conference on Machine Learning (ICML)*.

John L. Austin. 1962. *How to Do Things with Words*. Clarendon Press.

Simon Blackburn. 2016. *The Oxford Dictionary of Philosophy*. Oxford University Press.

F. H. van Eemeren and R. Grootendorst. 1984. *Speech acts in argumentative discussions: A theoretical model for the analysis of discussions directed towards solving conflicts of opinion.* Foris.

Steffen Eger, Johannes Daxenberger, and Iryna Gurevych. 2017. Neural End-to-End Learning for

Computational Argumentation Mining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11–22, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O K Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv*.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015. Towards Detecting Counter-considerations in Text. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 104–109. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2016. End-to-End Argumentation Mining in Student Essays. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1384–1394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Reed and Katarzyna Budzynska. 2011. How dialogues create arguments. In *Proceedings of the 7th Conference of the International Society for the Study of Argumentation (ISSA)*. SicSat.

Chris Reed, Katarzyna Budzynska, and Jacky Visser. 2016. IAT annotation guidelines for US2016. `http://arg.tech/US2016-guidelines`.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).

Sebastian Schuster, Joakim Nivre, and Christopher D Manning. 2018. Sentences with Gapping: Parsing and Reconstructing Elided Predicates. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1156–1168, New Orleans, Louisiana. Association for Computational Linguistics.

J. R. Searle. 1969. *Speech acts: An essay in the philosophy of language*. Cambridge University Press.

Christian Stab and Iryna Gurevych. 2014. Annotating Argument Components and Relations in Persuasive Essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510. Dublin City University and Association for Computational Linguistics.

Manfred Stede, Stergos Afantenos, Andreas Peldszus, Nicholas Asher, and Jérémy Perret. 2016. Parallel Discourse Annotations on a Corpus of Short Texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Jacky Visser, Barbara Konat, Rory Duthie, Marcin Koszowy, Katarzyna Budzynska, and Chris Reed. 2019. Argumentation in the 2016 US presidential elections: annotated corpora of television debates and social media reaction. *Language Resources and Evaluation*.

Justine Zhang, Arthur Spirling, and Cristian Danescu-Niculescu-Mizil. 2017. Asking too much? The rhetorical role of questions in political discourse. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1558–1572, Copenhagen, Denmark. Association for Computational Linguistics.

Top 1-8	Top 9-16	Top 17-24
, (31%)	– (2%)	or (1%)
and (12%)	, because (1%)	? (1%)
NONE (10%)	-lrb- (1%)	. and (1%)
, and (9%)	, which (1%)	to (1%)
, but (4%)	; (1%)	as (1%)
. (3%)	... (1%)	, so (1%)
because (2%)	- (1%)	that (1%)
but (2%)	when (1%)	if (0%)

Table 9: Words that separate two annotated locutions that overlap with one predicted locution. NONE indicates that the locutions are back-to-back without any separator.

Regex	Prec	Recl	F1
`(?<!i )said (\S + ){3,}`	.387	.295	.335
` ^'' `	.559	.119	.196
` , '' `	.552	.115	.190
`you say`	.500	.068	.120
`said that (\S + ){3,}`	.394	.047	.084
` : '' `	.875	.025	.049
`said ''`	.714	.018	.035
`called (\S + )+''`	.556	.018	.035
`blamed`	.500	.014	.028
`argued that (\S + ){3,}`	1.000	.004	.007
`insisted that (\S + ){3,}`	1.000	.004	.007
All above	.442	.590	.505

Table 10: Accuracy of reported speech detection.

Tregex	F1	Coverage
`S $ (VBD < said)`	.234	5%
`S $ (VBD < /says?/)`	.184	0%
`S $ (VBD < /claim.*/)`	.184	0%
`S $ (VBD < /argue.*/)`	.184	0%
`S $ (VBD < /insist.*/)`	.184	0%
` S < '' `	.352	15%
`S $- /:/`	.220	5%
All above	.395	23%

Table 11: Regular expressions (Tregex syntax) for speech identification and their accuracy. The first five patterns represent a clause that is a sibling of *said*, *say* or *says*, *claim.**, *argue.**, and *insist.**, respectively. The last two patterns represent a clause that includes an opening quotation mark and follows a colon, respectively.

Regex	Prec	Recl	F1	Regex	Prec	Recl	F1
\?	.751	.938	.834	^should	.800	.014	.028
^do	.485	.087	.147	^would	.538	.012	.024
^how	.759	.078	.141	^will	1.000	.011	.021
^what	.462	.064	.112	^was	.667	.011	.021
^is	.775	.055	.102	^where	.714	.009	.017
^why	.423	.039	.071	^when	.071	.009	.016
^did	.842	.028	.055	^which	.286	.007	.014
^are	.800	.021	.041	^have	.500	.005	.011
^who	.706	.021	.041	^were	1.000	.004	.007
^can	.611	.019	.038	^could	.182	.004	.007
^does	.588	.018	.034	^has	.333	.002	.004
All	.588	.972	.733				

Table 12: Accuracy of question detection.

A Module: LocutionExtraction

For R-CRF, we used sklearn-crfsuite 0.3.6. We conducted grid search, exploring all combinations of the bias feature ($\{1, 0\}$) and the following optimization parameters:

- Gradient descent using the L-BFGS method
  - L1 regularization: 0, 0.05, 0.1
  - L2 regularization: 0, 0.05, 0.1
- Passive Aggressive (PA)
  - Aggressiveness parameter: 0.5, 1, 2

For BiLSTM-CRF, we used the following parameter values:

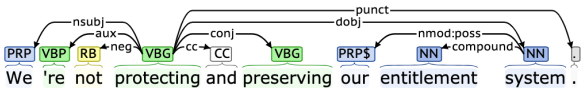
- BiLSTM hidden dim: 128
- Optimizer: Adam
- Learning rate: 0.001

For Ajjour, we used the following parameter values:

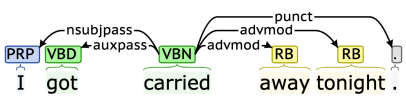
- Encoder BiLSTMs hidden dim: 128
- Output BiLSTM hidden dim: 5, 10, 20
- Optimizer: Adam
- Learning rate: 0.001

B Module: SubjectReconstruction

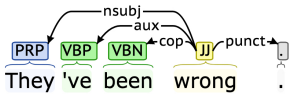
Conjunct (conj): Two verbs that are conjoined by a conjunction are likely to have the same subject. In the following example, *preserving* has the same subject as *protecting* does.



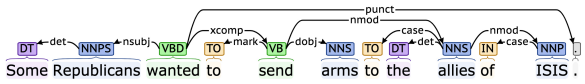
Auxiliary, passive auxiliary (aux, auxpass): An auxiliary verb that modifies a (passive) verb is likely to have the same subject as the modified verb does. In the following example, *got* has the same subject as *carried* does.



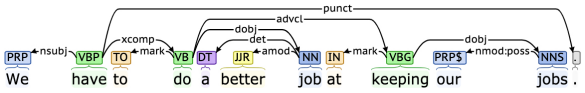
Copula (cop): A copula that joins a verb with its subject is likely to have the same subject as the verb. In the following example, *'ve* has the same subject as *wrong* does.

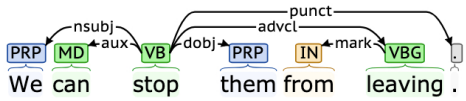


Open clausal complement (xcomp): An open clausal complement of a verb is likely to have the same subject as the verb does. In the following example, *send* has the same subject as *wanted* does.

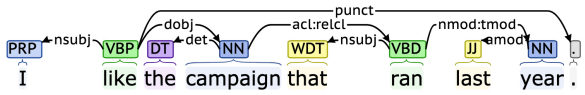


Adverbial clause modifier (advcl): An adverbial clause modifier of a verb may or may not have the same subject as the verb does. In the following examples, the two sentences have the same structure of verb + object + marked adverbial clause modifier. However, in the first sentence, *keeping* has the same subject as *do* does, whereas in the second sentence, *leaving* has a different subject than *stop* does. For reliability, we do not include adverbial clause modifiers for tracing a subject.





Relative clause modifier (acl:relcl): Sometimes a verb's direct subject is a relative pronoun, in which case we move to the word that is modified the current verb. In the following example, *ran* modifies *campaign*, which is the proper subject.



However, *which* may often refer to a phrase or a clause, and this method may not be able to capture that.

C Module: Revision

C.1 Copy Model

Suppose an input text is a sequence of words $w_1^E, \cdots, w_N^E$, and denote the word vector (e.g., word embedding) of w_i^E as $\boldsymbol{w}_i^E$. The BiLSTM encoder encodes each word $\boldsymbol{w}_i^E$ and outputs forward/backward hidden states $\overrightarrow{\boldsymbol{h}}_i^E$ and $\overleftarrow{\boldsymbol{h}}_i^E$ as

$$\overrightarrow{\boldsymbol{h}}_i^E, \overleftarrow{\boldsymbol{h}}_i^E = \overrightarrow{\text{BiLSTM}}(\boldsymbol{w}_i^E, \overrightarrow{\boldsymbol{h}}_{i-1}^E, \overleftarrow{\boldsymbol{h}}_{i+1}^E)$$
$$\overrightarrow{\boldsymbol{h}}_0^E = \overleftarrow{\boldsymbol{h}}_{N+1}^E = \boldsymbol{0}.$$

For the jth word to be generated, the LSTM decoder first encodes the concatenation of the previously generated word $\boldsymbol{w}^D_{j-1}$ and context vector $\bar{\boldsymbol{h}}^E_{j-1}$ (explained below), along with the previous hidden state as

$$\boldsymbol{h}^D_j = \text{LSTM}([\boldsymbol{w}^D_{j-1}; \bar{\boldsymbol{h}}^E_{j-1}], \boldsymbol{h}^D_{j-1})$$
$$\boldsymbol{h}^D_0 = [\overleftarrow{\boldsymbol{h}}^E_1; \overrightarrow{\boldsymbol{h}}^E_N].$$

Next, the decoder attends to the encoder's hidden states using an attention mechanism. The attention weight of the ith hidden state is calculated as the dot product of the hidden states from the encoder and decoder:

$$a_{ji} = \boldsymbol{h}^D_j \cdot [\overleftarrow{\boldsymbol{h}}^E_i; \overrightarrow{\boldsymbol{h}}^E_i], \hat{a}_{ji} = \frac{\exp(a_{ji})}{\sum_{i'} \exp(a_{ji'})}$$
$$\bar{\boldsymbol{h}}^E_j = \sum_i a_{ji} [\overrightarrow{\boldsymbol{h}}^E_i; \overleftarrow{\boldsymbol{h}}^E_i].$$

The probability of the ith input word being copied is proportional to the attention weight of the ith hidden state. On the other hand, calculation of the probability of newly generating the vth word in the vocabulary follows the standard attention decoder mechanism. Denoting these probabilities as $P_C(w_v)$ and $P_G(w_v)$, respectively, they are calculated as

$$P_C(w_v) = \sum_{i=1}^{N} \hat{a}_{ji} I(w^E_i = w_v)$$
$$P_G(w_v) = \text{softmax}(W_G[\boldsymbol{h}^D_j; \bar{\boldsymbol{h}}^E_j] + b_G)_v,$$

where W_G and b_G are corresponding weight matrix and bias vector. The final probability of w_v being generated is a weighted sum of $P_C(w_v)$ and $P_G(w_v)$, where the weight δ is automatically calculated as

$$\delta_j = \sigma(W_\delta \boldsymbol{h}^D_j + b_\delta)$$
$$P(w_v) = \delta P_C(w_v) + (1 - \delta) P_G(w_v),$$

where W_δ and b_δ are corresponding weight matrix and bias vector. The original method for calculating the weight (Gu et al., 2016) and a constant weight did not perform well on our task.

Beam search is used to choose the best output. Gradient clipping is used to avoid the exploding gradient problem.

C.2 Model Parameters

We explore the combinations of the following parameter values:

- Encoder hidden dim: 96, 128, 160, 192 (attention model) / 128, 192 (copy model)
- Beam size: 4
- Optimizer: Adam
- Learning rate: 0.001
- Gradient clipping: 1